ฝึกฝน dBASE

มารู้จักกับดีเบสทรีคอมพายเลอร์

ยืน ภู่วรวรรณ*

เมื่อดีเบสได้รับความนิยมและเป็นที่ชื่นชอบของผู้ใช้ ก็มีผู้เขียนโปรแกรมสำเร็จรูปใช้ร่วมกับดีเบสกันมากมาย และโปรแกรมหนึ่งที่ทำให้ดีเบสมีผู้ชื่นชอบขึ้นอีกมากคือ **คลิปเปอร์ ดีเบสคอมพายเลอร์** คลิปเปอร์เป็นชื่อทางการค้าของบริษัท Nantucket, Inc. คลิปเปอร์เป็นโปรแกรมคอมพายเลอร์ที่จะแปลงโปรแกรมดีเบสทรีให้เป็นโปรแกรม .EXE โดยเป็นโปรแกรมอิสระที่ไม่ต้องขึ้นกับดีเบสอีกต่อไป ความเร็วของการทำงานจะทำงานได้เร็วกว่าดีเบสมาก

จุดเด่นของการใช้คลิปเปอร์

- โปรแกรมทำงานได้รวดเร็วมาก
- สามารถปกป้องซอร์สโค้ด เพราะโปรแกรมที่ใช้เมื่อผ่านการคอมพายล์แล้วจะได้เป็น .EXE ซึ่งไม่สามารถมองเห็นซอร์สโค้ดได้
- คำสั่งที่ใช้ส่วนใหญ่ใช้ได้เหมือนดีเบสทรีทุกประการ ยกเว้นเพียงบางคำสั่งเท่านั้นเองที่มีการใช้ต่างกัน
- สามารถเชื่อมโยงกับโปรแกรมภาษาอื่นได้ง่ายกว่าดีเบสทรีมาก

* ห้องปฏิบัติการวิจัยไมโครคอมพิวเตอร์
ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
จุฬาลงกรณ์มหาวิทยาลัย

ขีดความสามารถของคลิปเปอร์

คลิปเปอร์มีขีดความสามารถบางประการที่เหนือกว่าดีเบสทรี (อย่างไรก็ตามในดีเบสทรีพลัส ก็มีคำสั่งบางคำสั่งที่เหนือกว่าคลิปเปอร์) คำสั่งของคลิปเปอร์มีลักษณะการเขียนเช่นเดียวกับดีเบส โดยใช้เอดิเตอร์เขียนโปรแกรม แล้วนำมาคอมพายล์ได้

ขีดความสามารถของคลิปเปอร์

- *มีคำสั่ง INKEY (ดีเบสทรีพลัส ก็มีแล้ว)*
- *ใช้ตัวแปรได้ถึง 64,000 ตัว ซึ่งมากกว่าดีเบสมาก*
- *ใช้จำนวนฟิลด์ได้ถึง 1,024 ฟิลด์*
- *สามารถสร้างฟังก์ชันขึ้นใช้เองได้*
- *สามารถสร้างความสัมพันธ์ระหว่างแฟ้มโดยมีแฟ้ม parent มีความสัมพันธ์กับ child*
- *สามารถเชื่อมโยงกับโปรแกรมภายนอกได้ทั้งภาษาแอสเซมบลี ซี ปาสคาล ฯลฯ*
- *สามารถใช้ฟิลด์เมโมในโปรแกรมได้ โดยอ่านข้อมูลมาที่ตัวแปรหรือส่งข้อมูลจากตัวแปรเข้าสู่เมโมฟิลด์*
- *ใช้แมคโครใน DO WHILE*
- *ใช้รีเคอร์ซีฟหรือแมคโครซ้อน ๆ กันได้*
- *วาดรูปสี่เหลี่ยมหรือตีกรอบได้*
- *สามารถสร้างแฟ้มข้อมูลขึ้นใหม่ในโปรแกรมได้*
- *มีคำสั่ง FOR...NEXT เหมือนเบสิก*
- *เพิ่มขีดความสามารถในคำสั่ง PICTURE และ RANGE*
- *มีฟังก์ชันในการหาชื่อฟิลด์ในไฟล์ หาค่าคีย์ที่กด หาจำนวนเรคอร์ด หาค่าหมายเลขบรรทัดของโปรแกรม หาชื่อโปรแกรมหรือกระบวนความที่กำลังทำงาน*
- *สามารถสร้าง HELP.PRG เพื่อให้ผู้ใช้ออกแบบระบบขอความช่วยเหลือได้*
- *เชื่อมโยงกับ OS ได้โดยตรง*

เมื่อเริ่มทดลองใช้คลิปเปอร์

เมื่อเขียนโปรแกรมด้วยดีเบสทรีเสร็จเรียบร้อยแล้ว และลองรันดูด้วยดีเบสจนเห็นว่ารันได้ถูกต้องแล้ว และต้องการคอมพายล์ เราเริ่มด้วยการใช้โปรแกรมคลิปเปอร์ โดยดูในแผ่นดิสก์ด้วย DIR จะพบ CLIPPER.COM เราจะเริ่มคอมพายล์ด้วย

C>CLIPPER<d:filename>

d: คือดิสก์ไดรฟ์ซึ่งเป็นไดรฟ์ที่โปรแกรมอยู่

การคอมพายล์จะเริ่มจากโปรแกรมหลักโปรแกรมแรก และหากโปรแกรมดังกล่าวเรียกโปรแกรมอื่น คอมพายเลอร์จะเรียกเข้ามาคอมพายล์ร่วมด้วยจนหมด

และหากพบบรรทัดใดผิด คอมพายเลอร์จะบอกรายละเอียดข้อผิดพลาดไว้ และหากไม่มีข้อผิดพลาดให้ลองขอดู DIR จะพบไฟล์ filename.obj ซึ่งเป็นออบเจคไฟล์ที่เกิดขึ้นจากการคอมพายล์

โปรแกรม .obj ที่เกิดขึ้นยังใช้งานอะไรไม่ได้ จำเป็นจะต้อง LINK โดยใช้ PLINK 86 ซึ่งมีอยู่ในแผ่นของคลิปเปอร์แล้ว ลองดูด้วย DIR ดู วิธีการลิงค์ขั้นพื้นฐานคือ การ LINK .obj ที่ผ่านการคอมพายล์แล้ว มาลิงค์ร่วมกับไลบรารี่ CLIPPER.LIB การลิงค์ทำได้ด้วยคำสั่ง

C>PLINK86 FILE<filename> LIB\CLIPPER\CLIPPER

การลิงค์นี้ จะใช้เวลานานพอควร โดยจะต้องรอและห้ามถอดแผ่นดิสเกตต์ออก และที่สำคัญจะต้องมีพื้นที่บนดิสก์พอเพียงสำหรับการสร้างโปรแกรม .EXE

อนึ่งการใช้ PLINK 86 นี้สามารถสร้างแฟ้ม .EXE ร่วมกับโอเวอร์เลย์ได้อีกด้วย วิธีการสร้างโอเวอร์เลย์ให้ดูจากคู่มือ

นอกจากนี้ ถ้าต้องการลิงค์แล้วให้ทำการร่วมกับ .obj ที่มาจากภาษาแอสเซมบลี ซี หรือภาษาอื่น โดยใช้ LINK .EXE ที่ใช้ในดอสได้เลย การลิงค์ในกรณีนี้ทำได้ดังนี้

C>LINK
object Modules [.OBJ]:<filename>

ให้กรอกชื่อแฟ้ม ถ้าต้องการลิงค์ .obj หลายตัวให้ใช้ชื่อไฟล์ + กัน เช่น MYFILE+CFILE+ASMFILE เมื่อกดรีเทอร์นจะปรากฏ

Runfile [filename.EXE]:<filename>

ให้ใส่ชื่อไฟล์ที่เป็นไฟล์ .EXE กดรีเทอร์นแล้วจะปรากฏ

List file [NUL.MAP]:<filename>

ลักษณะของฐานข้อมูล	คลิปเปอร์	ดีเบสทรี
จำนวนตัวแปร	64000	256
จำนวนฟิลด์ต่อไฟล์	1024	128
การใช้ Relation ในคำสั่ง SET RELATION TO	1 ไฟล์ เชื่อมกับไฟล์ลูกได้ 8 ไฟล์	1 ไฟล์ เชื่อมกับไฟล์ลูกได้ 1 ไฟล์

ตารางที่ 1 เปรียบเทียบความคล่องตัวระหว่างคลิปเปอร์กับดีเบส

ถ้ากดรีเทอร์นจะหมายถึงไม่ต้องการลิสต์ไฟล์ ถ้าต้องการก็ให้ใส่ชื่อไฟล์ที่ต้องการ เมื่อกดรีเทอร์นแล้วจะปรากฏ

libraries [.LIB]:CLIPPER

กดรีเทอร์น แล้วรอจนกระทั่งกลับมาที่โอเอสอีกครั้ง

ดีเบสทรีกับคลิปเปอร์ : ส่วนที่ต่างกัน

ปกติโปรแกรมที่เขียนด้วยดีเบสทรีมักจะนำมาคอมพายล์โดยคลิปเปอร์ได้ โดยจะมีเพียงบางคำสั่งเท่านั้นที่ใช้ไม่ได้ แต่ก็มักจะเลี่ยงออกได้เช่นกัน ลองดูการใช้คลิปเปอร์แล้วจะใช้อะไรได้เพิ่มเติมอีกบ้าง

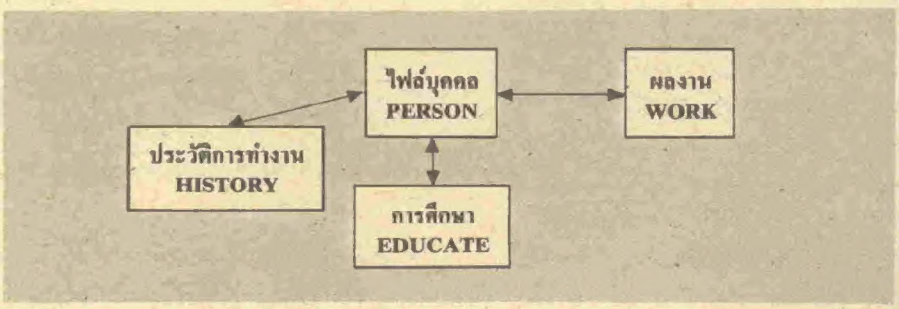
คลิปเปอร์ให้ความคล่องตัวเพิ่มขึ้น

คลิปเปอร์ให้ความคล่องตัวในเรื่องจำนวนตัวแปรและจำนวนฟิลด์ ซึ่งผู้ใช้จะหมดกังวลในเรื่องของตัวแปรและฟิลด์ได้เลย ดังรายละเอียดในตารางที่ 1 นอกจากนี้คลิปเปอร์ยังให้ความสัมพันธ์ระหว่างแฟ้มในรูปแบบพอยน์เตอร์ระหว่างไฟล์ได้ดี

สำหรับเรื่องของตัวแปรและชื่อฟิลด์นั้น ผู้ใช้คงไม่มีปัญหาเพราะสามารถใช้ได้โดยตรง แต่การเชื่อมโยงระหว่างแฟ้มด้วยฟิลด์พอยน์เตอร์คลิปเปอร์ยอมให้เชื่อมโยงได้ถึง 8 แฟ้ม ดังรูปที่ 1

โดยลองพิจารณาการใช้โดยการสร้างการเชื่อมโยงของแฟ้มทั้ง 4 โดยมีฟิลด์ PER_ID เป็นตัวเชื่อมโยง

```
SELECT 4
USE WORK
SELECT 3
USE EDUCATE
SELECT 2
USE HISTORY
SELECT 1
USE PERSON
```



รูปที่ 1

```
SET RELATION TO
 PER_ID INTO WORK;
 PER_ID INTO EDUCATE;
 PER_ID INTO HISTORY
```

จากตัวอย่างนี้ ถ้าเลื่อนเรคอร์ดพอยน์เตอร์ในไฟล์ PERSON เรคอร์ดพอยน์เตอร์ของไฟล์อื่นจะเลื่อนไปชี้ตาม

การ SAVE SCREEN และ RESTORE SCREEN

ผู้เขียนเคยเขียนโปรแกรม RESTORE SCREEN และ SAVE SCREEN ในรูปแบบของภาษาแอสเซมบลี และใช้กับดีเบสทรีพลัส ในวารสารไมโครเล่มที่ 26 มาแล้ว แต่สำหรับคลิปเปอร์นั้น มีคำสั่งนี้ให้แล้ว

การให้ความช่วยเหลือในโปรแกรม ในกรณีนี้ สามารถใช้การส่งผ่านพารามิเตอร์ก็ได้ โดยรูปแบบการขอความช่วยเหลือในคลิปเปอร์จะใช้คีย์ F1 เป็นตัวขอความช่วยเหลือ ผู้เขียนโปรแกรมสามารถเขียนโปรแกรม HELP.PRG ไว้ด้วย ในขณะคอมพายล์ และเมื่อผู้ใช้กดคีย์ F1 โปรแกรมจะวิ่งไปรันที่ HELP.PRG โดยอัตโนมัติ แต่ต้องแน่ใจว่าได้ LINK โปรแกรม HELP.PRG ไว้ด้วย

เมื่อกดคีย์ F1 คลิปเปอร์จะมองหาและส่งพารามิเตอร์เหล่านี้ให้ คือ

1. call_prg - เป็นตัวแปรที่เก็บตัวอักษรที่จะเก็บค่าชื่อของตัวแปรที่เรียกมายังโปรแกรม HELP.PRG

2. line_num - เป็นตัวแปรเก็บตัวเลขหมายเลขบรรทัดของโปรแกรมที่เรียกมายังโปรแกรม HELP.PRG

3. input_var - เป็นตัวแปรตัวอักษรที่จะเก็บชื่อของตัวแปรที่จะส่งผ่านเข้ามาขณะรออินพุท (โปรแกรมหยุดรอรับคีย์ ขณะอินพุท เช่น GET MEMVAR ก็จะส่ง MEMVAR ไปเมื่อกด F1)

โดยปกติเมื่อกด F1 แล้วโปรแกรมจะแสดงโดยวิ่งเข้ามาใน HELP.PRG และคลิปเปอร์แนะนำว่า ไม่ควรใช้รีเคอร์ซีฟใน HELP นี้ โดยแทรกโปรแกรมนี้ลงไป คือ

```
IF call_prg = "HELP"
  RETURN
ENDIF
```

ลองดูตัวอย่างโปรแกรมที่ 1 HELP.PRG ที่จะใช้สำหรับการขอความช่วยเหลือ

นอกจากนี้ เรายังใช้คำสั่ง SAVE SCREEN และ RESTORE SCREEN ใน HELP.PRG โดยสามารถสร้าง HELP ได้เหมือนกับโปรแกรมสำเร็จรูปอื่น ๆ ได้อย่างดี ลองดูวิธีการสร้างสกรีนขอความช่วยเหลือจากโปรแกรม

1. ในโปรแกรมหลัก เรานิยาม PUBLIC ในตัวแปรบางตัว เช่น ในที่นี้ใช้ help_code เป็นตัวแปรโกลบอลที่จะส่งต่อไปให้โปรแกรม HELP.PRG ใช้ได้ โดยโปรแกรมหลักเราเขียนได้ดังตัวอย่างในโปรแกรมที่ 2

```
* Sample Context-Specific HELP .PRG fro Clipper Compiler
*
* When the F1 function key is pressed while awaiting the
* input of a variable, Clipper invokes the help facility as
* though the command DO HELP WITH call_prg, line_num,
* input_var had been used. Clipper supplies the name of the
* calling program and the name of the variable that is
* awaiting input.
*
* Make parameters available to HELP .PRG before proceeding.
PARAMETERS call_prg, line_num, input_var
*
DO CASE
   CASE          call_prg   = "CUSTMAST";
        .AND. line_num  = 125;
        .AND. input_var = "LOANS"
   @ 23,5 SAY "Please enter interest for loantype & loantype"
   CASE          call_prg   = "CUSTMAST";
        .AND. line_num  = 231;
        .AND. input_var = "ACCOUNT"
   @ 23,5 SAY "Please enter the interest payments this month"
ENDCASE
RETURN
```

โปรแกรมที่ 1 **HELP.PRG**

```
* Program CUST_SVC to demonstrate the use of help_code
*
* Assign help_code value for the first program section
help_code = "01"
< program source code statements (DO, IF, CLEAR, etc.)>
* Assign help_code value for next section of program
help_code = "02"
<program source code statements (DO, IF, CLEAR, etc.)>
* etc., etc.
```

โปรแกรมที่ 2 กำหนดรหัส **help_code**

```
* Sample generalized HELP .PRG for Clipper.
*
* Assumes PUBLIC variable help_code has been defined in
* the main application program.
*
* NOTE: In this example, use of the commands SAVE SCREEN
* and RESTORE SCREEN eliminates the necessity of
* repainting the original input screen after the user has
* viewed the help information.
PARAMETERS call_prg, line_num_ input_var
*
* Parameters must be passed to the memory variables. Even
* if the parameters are not used by the program, they must be
* specified but may be called Dummy_1, Dummy_2, Dummy_3
IF help_code = "01"
  SAVE SCREEN
  * Erase without clearing GET's...
  @ 0,0 CLEAR
  ? "              AMORTIZATION UPDATES"
  ? "                   Procedures"
  ?
  ? "To see how a change in the loan terms will affect"
  ? "amortization, or to actually enter a change,"
  ?
  ? "Enter LOAN NO. to display current terms of loan"
  ?
  ? "Enter OPTION NO. indicating terms to be changed,"
  ? "then enter the new term in the adjacent space."
  ? "1 for term of loan in months - Enter new months."
  ? "2 for interest rate - Enter new interest rate."
  ? "3 for principal due - Enter new P.B. amount."
  ?
  ? "Press:"
  ? "F6 key to see the results of the calculation."
  ? "F7 key to make the new terms permanent."
  ? "F8 key to EXIT"
  ?
  ? "Press any key to go back to AMORTIZATION SCREEN."
  WAIT
  RESTORE SCREEN
  RETURN
ENDIF
If help_code = "02"
  SAVE SCREEN
  @ 0,0 CLEAR
  ? "See page 2-343 of the Procedures manual"
  ? "or contact the Loan Service Supervisor
  WAIT
  RESTORE SCREEN
  RETURN
ENDIF
```

โปรแกรมที่ 3 HELP ตรวจสอบรหัส help_code

2. คราวนี้เรามาเขียนโปรแกรม HELP.PRG ขึ้น โดยเมื่อมีการกดคีย์ F1 มันจะวิ่งมาทำงานที่ HELP.PRG และโปรแกรม HELP จะตรวจสอบรหัส help-code ตัวอย่างโปรแกรม HELP ในโปรแกรมที่ 3

คลิปเปอร์ กับฟิลด์เมโม

ฟิลด์เมโมในดีเบสทรี เป็นเรื่องที่ดีที่ทำให้เราสามารถเก็บข้อมูลแบบไม่กำหนดความยาวของเรคอร์ดได้ ฟิลด์เมโมจึงเหมาะที่จะเก็บข้อมูลประเภทบรรยาย เช่น Abstract รายงานวิจัย, คำบันทึกการรักษาของนายแพทย์ ฯลฯ แต่อย่างไรก็ตาม การใช้งานเมโมนี้ ไม่สามารถใช้โปรแกรมเรียกดูได้ แต่เมื่อใช้คลิปเปอร์สามารถใช้โปรแกรมโดยการใช้ชื่อฟิลด์เมโมถ่ายมาในตัวแปรสตริงเพื่อจัดการ เช่น ฟิลด์ เมโม ชื่อ REMARK เราสามารถเขียนโปรแกรมเป็น

```
MEM_DISP = REMARK
? MEM_DISP
REPLACE REMARK WITH
MEM_DISP
```

คลิปเปอร์ กับแมคโคร

ในดีเบสทรี เราใช้แมคโครในการแทนค่า แล้วนำมาแทนในโปรแกรม แต่สำหรับคลิปเปอร์แล้ว มีข้อจำกัดใช้งานได้ไม่เหมือนดีเบสทรี โดยคลิปเปอร์จะยอมให้ใช้ได้ในขอบเขตจำกัด เช่น

```
ใช้ได้ในคลิปเปอร์
file = "ABC"
USE & file
comp = "nmbr2 > nmbr1"
LIST name, city FOR & comp
```

```
ใช้ไม่ได้
command = "USE ABC"
& command
comp = "name, city FOR
nmbr2 > nmbr1"
LIST & comp
```

ผู้ใช้สามารถกำหนดฟังก์ชันใช้เองได้

คลิปเปอร์มีฟังก์ชันภายใน เช่นเดียวกับดีเบสอยู่แล้ว เช่น BOF( ) EOF( ) แต่อย่างไรก็ดีคลิปเปอร์ก็ยังยอมให้ผู้ใช้สร้างหรือกำหนดฟังก์ชันขึ้นเอง เหมือนภาษาคอมพิวเตอร์อื่นคือสามารถสร้างฟังก์ชันได้เหมือน PROCEDURE ลองดูตัวอย่างในโปรแกรมที่ 4

คลิปเปอร์กับคำบางคำสั่งในดีเบสที่ใช้ไม่ได้

คลิปเปอร์ไม่สามารถใช้กับคำสั่งบางคำสั่งที่มีใช้ในดีเบสได้ ตัวอย่างคำสั่งที่

```
* FUNC.PRG
DO Main
RETURN
FUNCTION Max
PARAMETERS n1, n2
IF (n1 > n2)
  RETURN (n1)
ELSE
  RETURN (n2)
ENDIF
PROCEDURE Main
INPUT "Enter number 1:" to n1
INPUT "Enter number 2:" to n2
? Max (n1, n2)
RETURN
```

โปรแกรมที่ 4 ตัวอย่างการกำหนดฟังก์ชันขึ้นเอง

ดีเบสทรีใช้	คลิปเปอร์ใช้
DISPLAY [FOR<condition>]	DISPLAY<field list> [FOR<condition>]
LIST [FOR<condition>]	LIST<field list> [FOR<condition>]
AVERAGE [FOR<condition>]	AVERAGE<field list>TO<memvar list>[FOR<condition>]
SUM [FOR<condition>]	SUM<field list>TO<memvar list> [FOR<condition>]

ตารางที่ 2 เปรียบเทียบคำสั่งที่ต่างกันระหว่างดีเบสกับคลิปเปอร์

คลิปเปอร์คอมพายล์ไม่ได้มีดังนี้ **ASSIST** และ **HELP, BROWSE, CHANGE, EDIT, INSERT** และ **MODIFY COMMAND SET HELP, SET HEADING, SET MENUS SET SAFETY** และ **SET TALK** แต่บางคำสั่งที่มีการใช้งานในการดีบักหรือขอดูรายละเอียดเกี่ยวกับชื่อตัวแปรซึ่งคลิปเปอร์ไม่มีให้ใช้ซึ่งได้แก่

DISPLAY MEMORY และ **LIST MEMORY**

DISPLAY STATUS และ **LIST STATUS**

DISPLAY STRUCTURE และ **LIST STRUCTURE**

SET DEBUG SET ECHO และ **SET STEP**

บางคำสั่งคลิปเปอร์ใช้แตกต่างกันออกไป เช่น DISPLAY, LIST, AVERAGE หรือ SUM ดังตัวอย่างในตารางที่ 2

คลิปเปอร์กับอินเด็กซ์ไฟล์

การสร้างอินเด็กซ์ไฟล์ของคลิปเปอร์แตกต่างจากของดีเบสทรี ดังนั้นจึงไม่สามารถใช้ร่วมกันได้ การสร้างอินเด็กซ์ไฟล์ใด ๆ จะต้องสร้างด้วยตัวคลิปเปอร์เอง โดยอาจจะใช้เขียนเป็นโปรแกรม INIT.PRG เพื่อใช้ในการสร้างไฟล์ เมื่อสร้างไฟล์ด้วยคำสั่ง INDEX ON<ชื่อฟิลด์> TO<ชื่อไฟล์> แล้วจะได้ไฟล์ที่มี .NTX แทนที่จะได้ .NDX เหมือนของดีเบส

คลิปเปอร์กับคำสั่งพิเศษที่นอกเหนือจากดีเบส

ผู้สร้างคลิปเปอร์ได้พบกับจุดอ่อนของดีเบสทรีมาแล้ว จึงพยายามสร้างคำสั่งบางคำสั่งให้คลิปเปอร์ใช้งานได้สะดวกขึ้น ลองพิจารณาคำสั่งที่เพิ่มพิเศษมานี้ จะเห็นว่าคลิปเปอร์มีของดีเพิ่มขึ้นอีกหลายอย่างทีเดียว

@...BOX เป็นคำสั่งสำหรับการตีกรอบ โดยรูปแบบของคำสั่งเป็นดังนี้

@<t,l,b,r>BOX<string> โดย สตริงสุดท้าย เป็นการกำหนดตัวอักษรกราฟิก เช่นตัวอย่าง

```
frame = CHR(101) + CHR
        (205) + CHR(187)
        + CHR(186) ;
        + CHR(188) +
        CHR(204) + CHR
        (200) ;
        + CHR(186) +
        CHR(176)
@ 1, 0, 24, 79 BOX frame
```

และถ้าต้องการเคลียร์บล็อกนี้ก็ใช้

@ 1, 0, 24, 79 BOX " "

CREATE เป็นการสร้างแฟ้มข้อมูลขึ้นใหม่โดยใช้โปรแกรมช่วย หลักการสร้างแฟ้มจะสร้างโครงสร้างแฟ้มเหมือนกับการสร้าง extended file ในดีเบสทรีก่อนแล้ว create from ลองดูจากตัวอย่างในโปรแกรมที่ 5

CALL คำสั่งที่เชื่อมโยงกับ .obj ตัวอื่นซึ่งอาจจะเป็นภาษาแอสเซมบลีหรือซีก็ได้ รายละเอียดการเชื่อมโยงกับภาษาซี ผู้เขียนจะได้กล่าวถึงในโอกาสต่อไป

FOR..NEXT คำสั่งนี้เหมือนของภาษาเบสิกทุกประการ ลองดูจากโปรแกรมตัวอย่างนี้

```
FOR i = 10 TO 0 STEP-2
 ? i
NEXT
```

FUNCTION เป็นการสร้างฟังก์ชันขึ้นใช้เอง โดยรูปแบบของคำสั่งที่ใช้จะเป็น FUNCTION<name>.. RETURN <value>ตัวอย่างการใช้ดังโปรแกรมที่ 6

PUBLIC clipper คำสั่งนี้เป็นการสร้างให้ผู้เขียนโปรแกรมคลิปเปอร์สามารถนำเอาไปใช้ในดีเบสทรีได้ โดยไม่มีข้อผิดพลาดปรากฏออกมา เพราะสามารถตรวจสอบด้วย If clipper ได้ ดังตัวอย่าง

```
PUBLIC clipper
IF clipper
 @ 1, 0, 3, 10 BOX "*"
ELSE
 @ 1, 0 SAY "**********"
 @ 2, 0 SAY "*        *"
 @ 3, 0 SAY "**********"
ENDIF
```

RESTORE SCREEN เป็นคำสั่งที่จะให้คลิปเปอร์นำข้อมูลสกรีน (จอภาพ) ที่เก็บไว้มาแสดงผล ข้อมูลภาพที่เก็บไว้คือ SAVE SCREEN

SAVE SCREEN เป็นคำสั่งเก็บข้อมูลภาพข้อความที่แสดงบนจอไว้ เพื่อแสดงข้อความอื่นก่อน ประโยชน์เพื่อการใช้ HELP นั่นเอง

```
CREATE New_stru
DO WHILE .T.
   CLEAR
   APPEND BLANK
   @ 5,0 SAY "Field name:"    GET Field_name
   @ 6,0 SAY "Field type:"    GET Field_type
   @ 7,0 SAY "Field length:"  GET Field_len
   @ 8.0 SAY "Field decimals:" GET Field_dec
   READ
   IF  LEN(TRIM(Field_name)) = 0
      DELETE
      PACK
      USE
      EXIT
   ENDIF
ENDDO
CREATE Newfile FROM New_stru
```

โปรแกรมที่ 5 ตัวอย่างการใช้คำสั่ง CREATE

```
DO Get-string
RETURN
FUNCTION CENTER
PARAMETERS string
col_pos = LEN (string)
IF col_pos<80
  col_pos = INT ((80-col_pos)/2)
ELSE
  col_pos = 0
ENDIF
RETURN (col_pos)
PROCEDURE Get-string
ACCEPT "Enter character string to center:";
       TO center_it
CLEAR
@ 12, CENTER (center_it) SAY center_it
RETURN
```

โปรแกรมที่ 6 ตัวอย่างการใช้คำสั่ง FUNCTION

ส่งท้าย

รายละเอียดของคลิปเปอร์หาดูได้จากหนังสือคู่มือของคลิปเปอร์ ซึ่งผู้เขียนพยายามย่อมาจากหนังสือคู่มือดังกล่าว และหวังว่าผู้ที่ได้ลองใช้คลิปเปอร์แล้วจะต้องชอบใจ